INDEX

NAME: R. Sai Ganesh STD.: _____ SEC.: _____ ROLL NO.: 7 SUB.: _____

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
1.	24/7	Exploring Deep learning Platform.		
2.	07/8	Implement a classifier using open source dataset		
3.	14/8	Study of classifier with respect to statistical Parameter.		
4.	14/8	Build a simple feed forward neural network to register.		
5.)	2218/25	Study of activation function and its roles.		
6.)	9/9/25	Implement gradient descent & back Propagation in deep neural network.		
7.)	16/9/25	Build a CNN model to classify cat and Dog img.		

24/7

Exp-1

1) Exploring The Deep Learning Platform

Various deep Learning Platforms:-

Google Colab:-

→ Free online jupeter Note books with GPU/TPU.

→ Ideal for students, resources, hobbyists Accesible with google account.

Jupyter Notebook:-

→ Not a framework, but an interactive codeing environment.

→ Combines Codes, markdown, outputs.

→ Popular for Exploration, visualization tutorials.

Key Difference between Google Colab and Jupyter.....

Google Colab	Jupyter Notebook.
• Cloud based plateform.	• Local / browser-based.
• Build in free GPU / TPU	• GPU manual Setup.
• Runs in browser.	• Anaconda + Jupyter Installatio
• Easy sharing.	• Manual file sharing.
• Saves to google drive.	• Saves to local Syster

Key framework of Pytorch · Tensorflow.

Tensorflow :-

Creator / organization :- Google (2015).

Main features :- Scalable across CPUs, GPUs, High-performance model training Integrated Keras API for Simplicity visualization via Tensor Board.

Popular use Case :- Computer Vision.
NLP
ML

Pytorch :-

Organization :- Facebook AI Research [FAIR] (2016).

Main features :- Dynamic Computational graph.
Native Pythonic Syntax.
Strong GPU accelerating Support.

Popular Use Cases :- Research and Academic Projects
NLP models.
Fast model Prototyping.

Graph Type :- Dynamic.

7/8

Exp.2

Implement a classifier using open Source dataset

AIM:-

To build a machine learning classifier to predict the Species of Iris flowers using decision Tree Algorithm.

OBJACTIVE:-

- Understands and apply decision Tree classification on the Iris Datasets.
- Preprocess the dataset and splits it into training and testing sets.
- Train, the model and evaluate its performance using accuracy matrics.

Pseudo Code:-

1.) Import necessary Libraries.
2.) Load the Iris dataset.
3.) Standardize features.
4.) Split data into training and testing sets.
5.) Initialize Decision Tree classifier.
6.) Train the model on training data.
7.) Predict target on test data.
8.) Display results.
9.) END.

Source Code:-

```
Import numpy as np :
from sklearn.datasets import load_iris.
From sklearn.model_selection import train_test_split.
from sklearn.Preprocessing import standard scalar.
From sklearn.tree.import Decesion Tree classifier.
from sklearn.matrics import accuary_score, Procistion.
score.
    recall.score, ti_score, confusion_matrix

iris = load_iris().
x,y = iris.data, iris target.

Scalar = standard Scalar.
x_scaled = Scaler.fit_transform(x).

dt_model = Decision Tree classifier (random_state=42).
dt_model.fit ( x_train, y_train).
Y_Pred = dt_model.Predict (x_test).
accuracy = accuracy_score (y_test, y_Pred).
Precision = Precision_score (y_test, y_Pred, average='marco').
recall = recall_score (y_test, y_Pred, average='marco').
f1 = f1_score (y_test, y_Pred, average='marco').
Conf_matrix = Confusion_matrix (y_test, y_Pred):.
```

OBSERVATION :-

- The Decision tree classifier Succesfully learned Platforms from the training data & accurately Predicted the species of iris flowers on the test data.
- The model obtained a high accuracy, indicating that the dataset is well for decision tree classification.
- The Simplicity of the Iris dataset and clear Speration..
- The use of random state ensured reproducible results for evalutions.

Result :-

Decision tree classifier is successfully implement and texted using an open source Iris dataset.

Output :-

Accuracy : 1.0

	Precision	Recall	F1-Score	Support.
Setsa	1.00	1.00	1.00	10
Versicolor	1.00	1.00	1.00	9
Virginica	1.00	1.00	1.00	11
accuracy				
Macro avg	1.00	1.00	1.00	30
weight avg	1.00	1.00	1.00	30

14/8

Exp:3

Study of the classifiers with respect to statical Problems

Aim:-

To study and Compare the performance of different algorithms using statistical evaluation paramater such as accuracy, Percision, recall, Fi-sacre on a oper sane dataset.

objective:-

- To Select a hesitable oper-Source dataset for binary classifier.
- To ReProcess the dataset using techinques to improve model Performance.
- To implement and train classifiers.
  - → SVM.
  - → Decision Tree.
- To evaludte the models useing statical Parameter such as accuracy, decision, recall and fi-source.
- To analyse and Compare the Performance of each classifier based on the evaluation refulse.

Pseudo Code :-

1:) load digits dataset into features x and labed

- Split data into training (x-train, y-train).
- Initialize KNN classifier with a choosen K log. 3) and fit it on x-train and y-train.
- Predict labels on x-test, calculate accuracy complaining with y-test and print the result.

Logistic regression :-

- Load digits data into features . ,
- Split data into training (x-train, y. train). and testing (x-test, y-test) sets.
- Initialize logistic regression classifier. and train it on x-train and y-train.
- Predict labelson x-test. Calculate accuracy.

Result :-

The Successfully Implemented.

Exp 4

14/8

4. Simple Feed Forward Neural Network

AIM:-

To design and implement a simple feed forward Neural Network to classify Hand-written digits from MNIST dataset.

Pseudocode:-

- Load MNIST dataset.
- Pre Process data - Normalise all pixel values to range [0,1].
- Define Neural network architecture.

  Input layer : 784 Neurons.

  Hidden layer : 128 Neurons.

  Output layer : 10 Neurons.
- Configure Training Parameters.
  - loss.
  - optimizers.
- Train Neural Network.
- Evaluate the model using test dataset.

observation :-

- Dataset used - 6000 Training Image.
  1000 Testing Image.
  Gray scale.
- Input layer - 784 Neurons.
  Hidden Layer - 128 Neurons.
  output layer - 10 Neurons.

objective :-

- To load and Preprocess the MNIST dataset for neural network input.
- To build feed forward NN to hidden layers.
- To train model using gradient desc
- Predict the class.

Pseudo Code :-

BEGIN :-

Result :-

Simple Feel Forward Neural Network is Implemented.

Exp: 5 Study of activation function and its Roles.

Aim:-

To study different activation functions used in neural network and analyse the role.

Algorithm:-

Define Common activation functions:-

Sigmoid.

Hy Perbolic Tangent (Tanh).

Rectified Linear unit (ReLu).

leaky ReLu

Softmax.

- Generate a range of input values.
- Apply each activation function on the input.
- plot the outputs to study their characteristics

Testing :-

- For $x = 5$

  Sigmoid $\simeq 0.9933$.

  Tanh $\simeq 0.999$.

  Relu $= 5$

  Leaky Relu $= 5$.

For $x = [1, 2, 3]$

Softmax $= [0.09, 0.24, 0.66]$

Range :-

Sigmoid $- (0, 1)$.

Tanh $- (-1, 1)$

Relu $- [0, \infty]$

Leaky Relu $- (-\infty, \infty)$

Softmax $- (0, 1)$

OBSERVATION:-

Sigmoid maps inputs to range (0,1) useful for probabilities but suffers from vanishing.

Tanh maps input to range (-1,1)

ReLU outputs positive values directly and zero otherwise.

Leaky ReLU allows small negative values.

Softmax converts outputs into probability distribution.

RESULT:-

Therefore the study of activation function and its roles are completed successfully

PSEUDOCODE :-

BEGIN

Initialize weights and biases randomly

For each in range (max - epochs) :-

For each input sample :-

Forward pass

Compute $z = w * x + b$.

Apply activation to get A.

Compute output prediction y-pred.

Compute Loss.

Loss = Costly (y-true, y-pred).

Backward pass.

Compute gradients dw, db using chain rule

Update Parameters :

$w = w - \alpha * dw$.

$b = b - \alpha * db$.

END FOR.

Print loss after each epoch.

END FOR.

END.

9/9/25

Exp :6

Implement gradient descent & back propagation in deep neural network.

Aim:-

To implement gradient descent & back propagation in deep neural network.

OBJECTIVES:-

To understand the working of gradient descent. as an optimization method.

To implement back propagation for updating weights in a neural network.

To observe how loss decrease with iterations.

Output :-

- Epoch 1/4 :

loss : 0.5954 training : 0.5607.

Epoch 2/4 :

loss : 0.5543

OBSERVATION :-

→ loss decreases as the number of iteration increases.

→ Weights and biases adjust to minimise error.

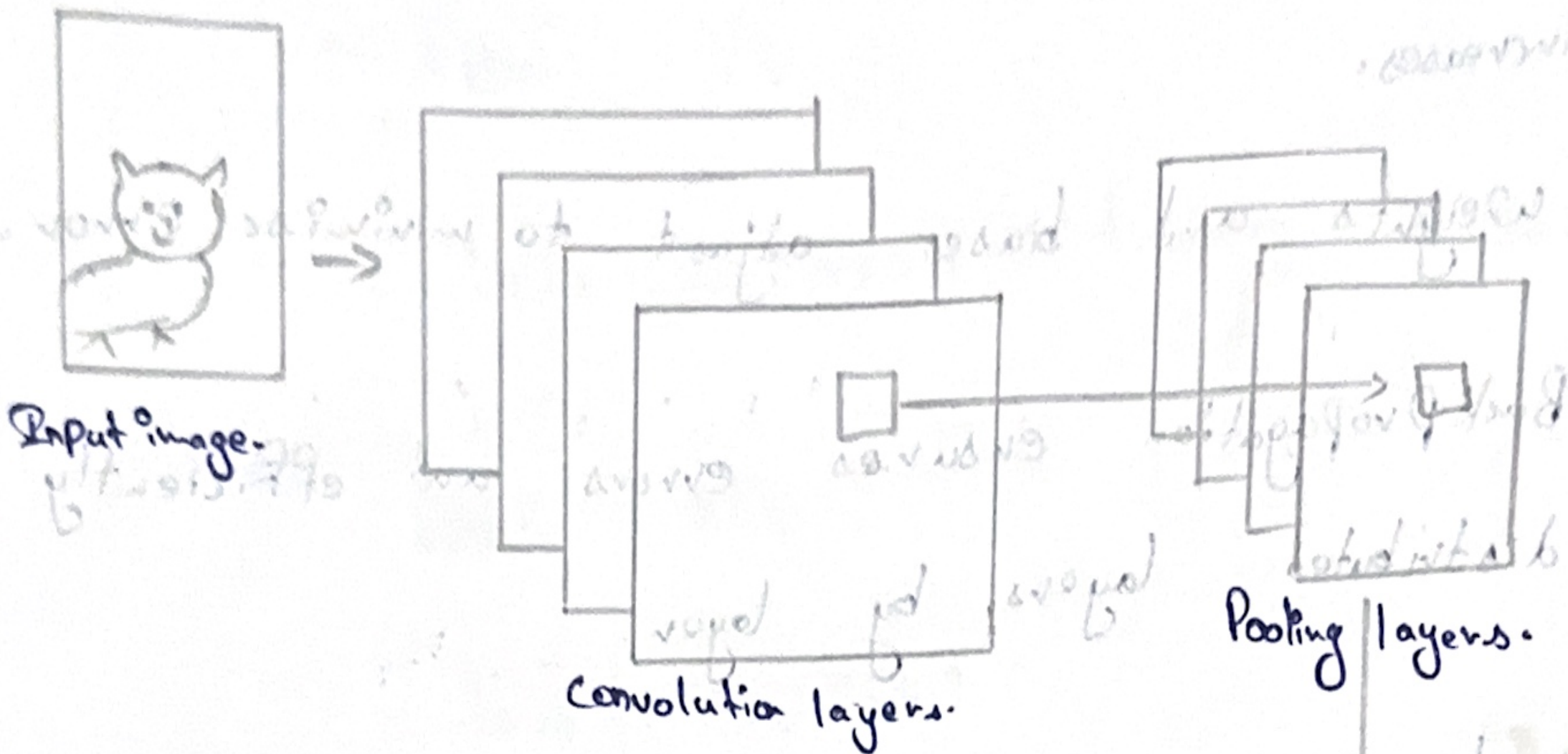
→ Back Propagation ensures errors are efficiently distributed layers by layer.

→ learning rate (η) greatly influences. Convergence Speed.

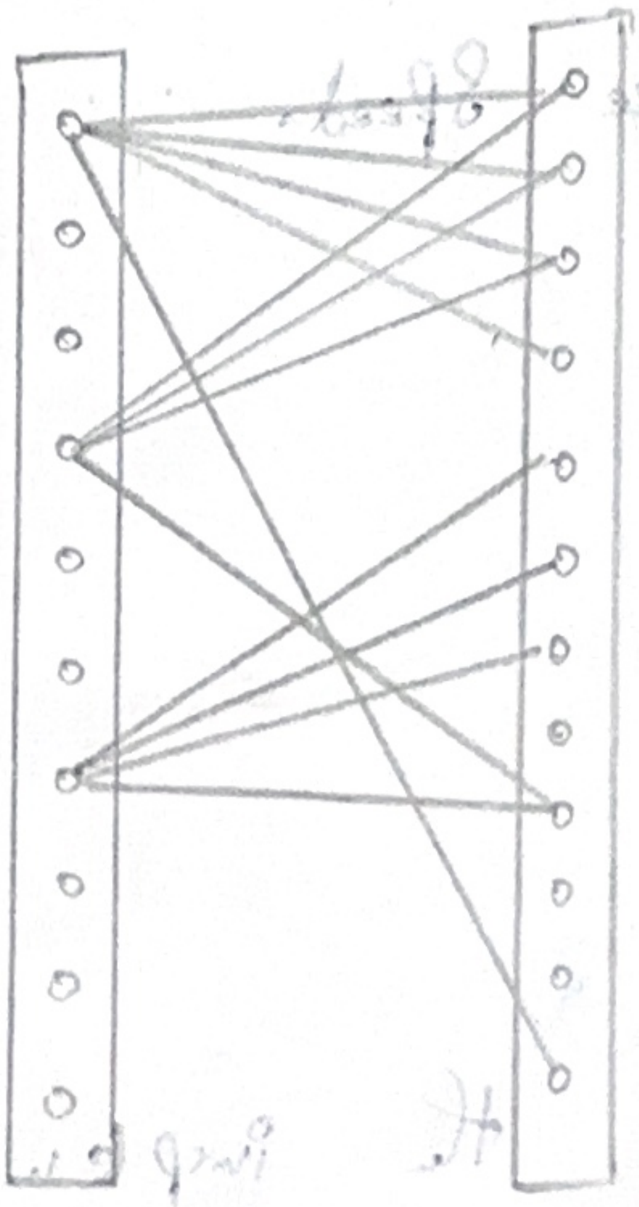
RESULT :-

Therefore the implementation of gradient descent and back Propagation in deep neural network.

Convolution + Pooling layers.

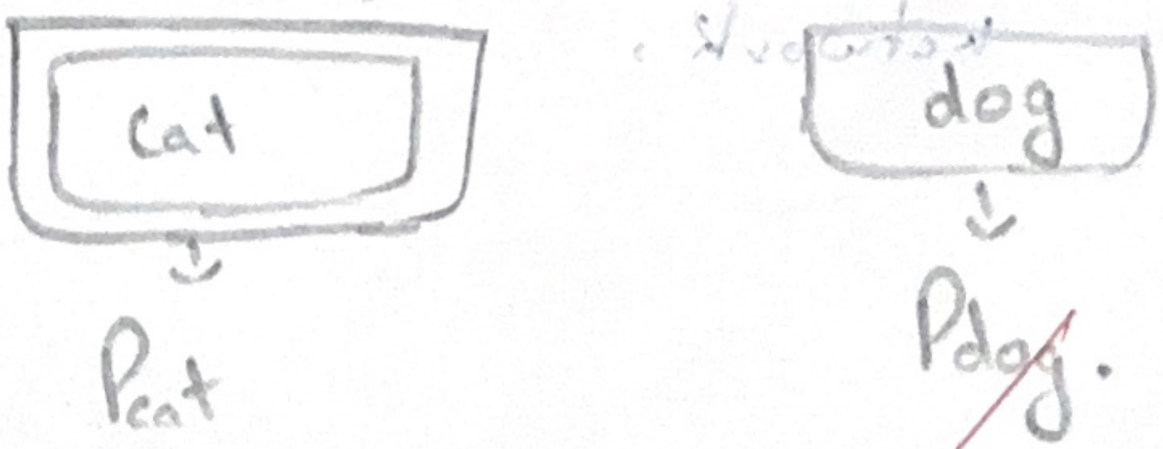


Fully connected layers.



vec

fully connected layer.



P_{cat} P_{dog}.

Output.

Nx binary classification.

16/9/25

Lab-7 : Build a CNN model To classify cat Dog/wag

Aim:- To build and train CNN model that classify images into 2 categories cat & Dog.

Objective:-

1.) To understand the architecture of Convolutional Neural Network.

2.) To PreProcess image dataset for training and testing.

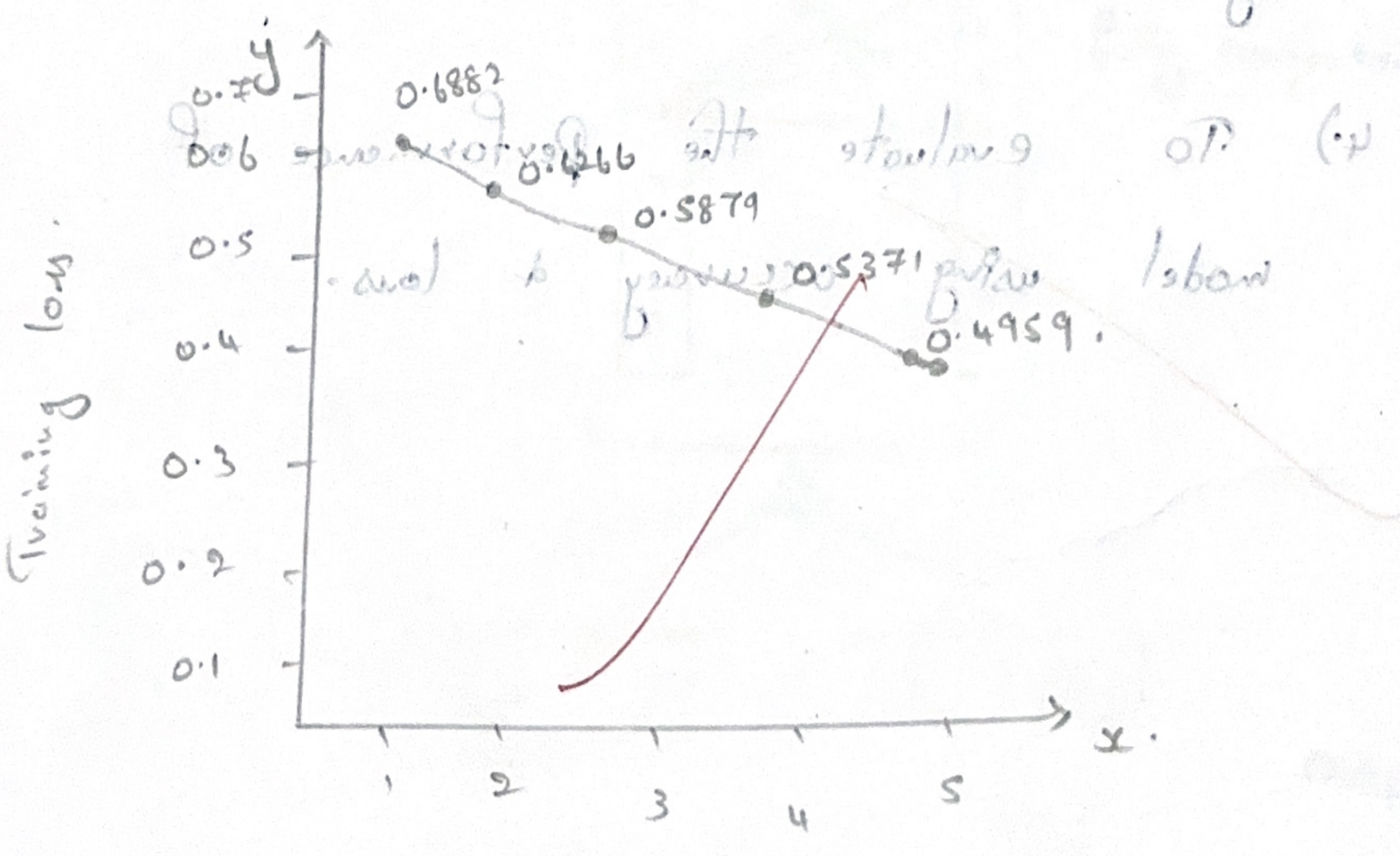
3.) To implement a CNN model using Py torch.

4.) To evaluate the performance of model using accuracy & loss.

CNN architecture.

Epoch	loss	Accuracy	validation
1	0.6882	55.70%	56
2	0.6266	64.30%	65
3	0.5879	69.20%	70
4	0.5371	74.25%	76
5	0.4959	76.25%	78

Graph:-



Pseudocode :-

BEGIN:-

1:) Import necessary libraries (Torch, Torch.nn).

2:) Load the dataset.

3:) Create a class CNN with.

- Convolutional layer + Relu + Max Pooling.
- Fully Connected layers.
- Output layer with 2 neurons.

4:) Train the data.

Initialize model, loss function

For each epoch :

For each batch in training data :-

Forward Pass.

Compute Loss.

Back Propagate loss.

update weights.

Print training loss.

5:) Testing.

- Evaluate model on Test data.
- Calculate accuracy.

END.

Observation :-

1.) Dataset.

- Dataset contains image of cats and dogs.

- Images resized to 128 x 128 & normalized before training.

2.) At begining of training, CNN had randomly intializing weights, which resulted in high training loss and low accuracy.

3.) As multiple epochs, CNN layer begins to extract meaningful features such as edges, Textures, shape.

4.) Polling layers helped reduce dimensionlity while retaining most important feature.

5.) After several epochs, models loss decreases while accuracy improves.

Result :-

Implemented - Build CNN model to classify Cat & Dog image.